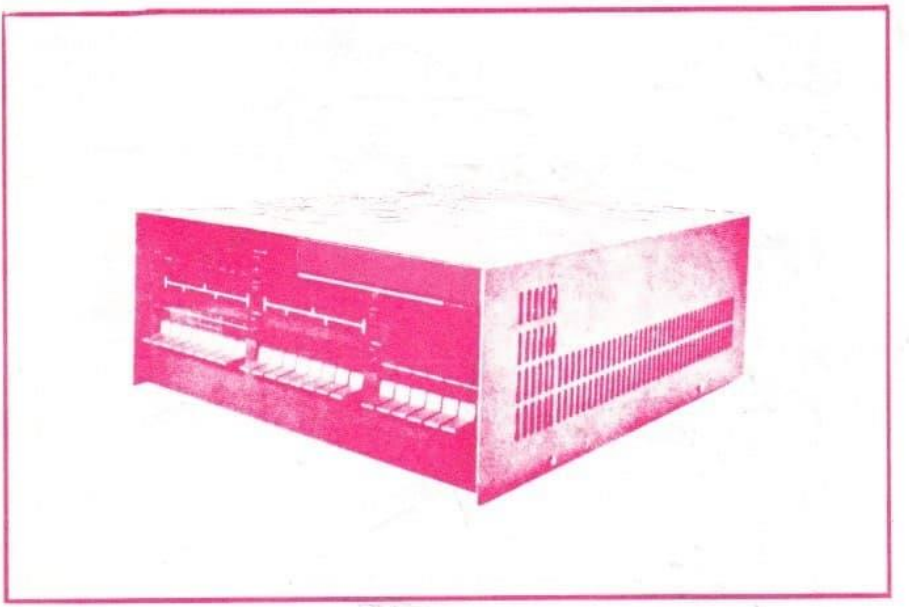
OTTAWA HOME COMPUTING

Feb. 17, 1986
Vol. 2, No. 8
$1.60

NOSTALGIA NIGHT



IMSAI's 8080 (Altair-compatible) *circa 1976*

INSIDE...

Editorial 3
February Agenda 4
Upcoming Meetings 5
Musical Adventures of a Tenderfoot 6
THIS MONTH'S THEME:
You've Come a Long Way, Micro 8
C-64 Game Design 10
Different Ways of Talking to a Microcomputer ... 12
January Minutes 14
Ramblings 15

C-64 DISK OF THE MONTH

The C-64 Disk of the Month is carried at the stores listed below. It sells for $4.50 and is available two days after each meeting.

A FLORIST & COMPUTERS	1152 Ogilvie Road	744-0220
COMPUCENTRE	Carlingwood Mall	729-0448
	Rideau Centre	230-3672
MEDIA LAND (G-Plus)	130 Albert Street	230-7750
	301 Moodie Drive	820-7326
MGI COMPUTER CORP.	1501 Carling Avenue	722-1000
MR DISKETTE	119 O'Connor Street	232-5203
SILICON ALLEY	79 Bank Street	232-2968
ZAP & ZOOM	435 Kent Street	232-4400

RENEWAL TIME

NAME _____

ADDRESS _____

CITY _____ PROV. _____

POSTAL CODE _____ PHONE _____

COMPUTER _____ OHC # _____

PLEASE MAKE CHEQUES PAYABLE TO: ($15/YEAR)
OTTAWA HOME COMPUTING
P.O. BOX 4164, STATION 'C', OTTAWA, ONTARIO, K1Y 4P3

OTTAWA HOME COMPUTING is the newsletter of the Ottawa Home Computing Club. Membership is open to all with a genuine interest in personal computing for $15/year in Canada. Membership includes OTTAWA HOME COMPUTING, which is published 10 times a year. Meetings are usually held on the third Monday of each month, 7:30 p.m. at Charlebois High School, corner of Heron Road and Alta Vista Drive in Ottawa.

When submitting articles, please type or write legibly on 8 1/2 by 11 inch paper, double spaced. Articles may also be submitted on disk, or in a "print-ready" format. Contact the editor for more information.

Permission to reprint articles appearing herein is hereby granted, provided credit to both the author and Ottawa Home Computing is given.

Advertisements of personal items up to 25 words for one insertion per issue are free to members. Inquire for commercial rates.

Address correspondence to:
Ottawa Home Computing Club,
P.O. Box 4164, Station "C",
Ottawa, Ontario K1Y 4P3

CLUB EXECUTIVE

President	
Paul Irwin	728-6728
Secretary/Treasurer	
Pierre Castricum	521-6840
Membership	
Mike Bryan	723-0404
Bulletin Board Editor	
Don White	829-2082
Newsletter Editor	
Bob Mason	837-3197
Apple Librarian	
Scotty Adams	225-0345
Commodore Librarian	
Paul Anderson	521-3116
Education Director	
Wayne Schaler	728-7811
Director at Large	
Marv Bero	225-8551
BULLETIN BOARD (not yet operational)	723-2252

OTHER RESOURCE PEOPLE

Retail Disk Distribution	
Peter Martin	731-7700
COMAL Librarian	
Ed Steenhorst	820-5389

Ottawa Home Computing Editor:
Bob Mason,
Box 183, R.R. 2,
Orleans, Ontario K1C 1T1

EDITORIAL
by Bob Mason

I think this is the best issue of the newsletter since I've been editor. There are three well-written articles by club members: Bob Handforth has some excellent tips for music programmers; John Batchelor discusses copy protection and software from his perspective as a programmer; and Patrick Yang compares the five most popular languages for the personal computer. We had to hold off Jim Sutton's review of Newsroom (done with Newsroom) until next month. Look for it!

As well, there's an article on the Nostalgia Night theme of the next meeting. I hope people will bring out their old equipment, books, magazines and posters (and anything else) and make this night really interesting.

I'm pleased with the way club members have responded to my pleas for help. Keep up the good work. But don't forget the other club activities.

I've heard criticism of the general meetings, criticism of the SIG meetings, criticism of the Disk of the Month (they don't tell me the complaints they have about the Newsletter, I guess). Talk to the executive. Make your feelings known. Maybe we need a Complaints Night, where members can put forth their suggestions. See the article on page 4 "Special Interest Groups R.I.P." Feedback!

AGENDA
February 17 Meeting

7:30 Disk of the Month
Membership Sales
Socialization

8:00 Business Meeting

8:30 Display:
NOSTALGIA NIGHT
Rooms 223 and 225
(see note in "Upcoming Meetings")

9:00 Special Interest Groups

Group	Room

Apple & Mac Room 227
Browse through the library, order disks, discuss applications, problems, etc. Now even easier to get to in a new location.

Commodore and Disk of the Month Room 219
Same as above, but for Commodore. If Paul Anderson doesn't have the answer for you, he'll direct you to someone who does.

COMAL Room 221
Comal is a programming language similar to BASIC, but much faster, more powerful, and easier to use. Find out about it.

Forth Room 230
Forth language is shorter and faster than BASIC. (20 to 600 times faster). Used with robots, sensors, and many appliances. Much more

Newsletter Room 210
The Newsletter editor and/or members of the Newsletter committee will be available for discussion, suggestions, submissions, whatever.

Beginning Computerists Rm 224
The lone survivor of the Special Interest Groups, this is for people who are just getting started, and have questions or just want to talk about computing.

Special Interest Groups R.I.P.

The passing was officially announced Jan. 20; the cause of death has been listed as a massive lack of interest. The body was on view in rooms 223 (Computer Graphics), 212 (Telecommunications), 227 (Data Base Users), 215 (Machine Language) and 225 (Music) -- no one attended. Resurrection is doubtful. Condolences to the father -- Wayne Schaler. His favorite son, Beginning Computerists, was found to be alive and well in Room 224.

* * * * *

But seriously folks...
I looked in on the 9-10 p.m. activities and found a well-attended and lively discussion in the Apple group -- Scotty, what's the secret? Commodore, COMAL and Forth were active on a lower level. Beginning computerists was attended by the regulars. The special tutorial on Relative and Sequential Files had a good number in attendance. The rest was silence.

The SIGs were well-attended initially, but after 2 or 3 meetings, started to drop off. The executive had hoped that these groups would be self-perpetuating, that they would determine a structure that would ensure their continuation. This did not work, and I think it didn't work because people were looking for a strong leader in each group -- someone to direct it, to teach.

The first few meetings were useful because they answered the questions of those in at-

tendance. But after that there was nothing new, and people left.

There are several club members who have a good knowledge in certain areas of computing. Some of these are on the Club executive, and already spending as much as 30-40 hours a month on club business. Others have heavy work commitments, often caused by that good knowledge. In other words, we do not have a vast pool of teachers able to give tutorials every month.

Where Do We Go From Here?

We do have a vast, largely untapped resource -- our members. I'm sure several could answer questions I have about Paperclip, about downloaded sequential files, about Babel fish in my ear, etc. I don't need a teacher, I need a 'rap' session.

What about you? What are your needs and how can the club best fulfill them. Talk to one of the executive members. Write a letter to the Newsletter. Tell us what you like and don't like, what you need and don't need, and how you can help us and how we can help you.

==========

UPCOMING MEETINGS

In the following list, the dates of meetings is firm but some topics are not.

Feb. 17 **Nostalgia Night**

Sometimes looking back helps you to see what's ahead, and in this young industry, it certainly is amazing to see the progress made in 12 years. So on this night we'll be looking back at some of the older systems. If you've got an old computer, or even a Video Game System, stashed in the closet, or even still in use, bring it out for others to see. Old books and posters would be interesting too. All material can be placed in rooms 225 and 223 during the meeting, and these rooms will be secured. If you're bringing in a system, give Wayne a call beforehand if you can. That way unnecessary duplication of efforts will be avoided.

Mar. 24 **Integrated Software**

Integrated software incorporates a combination of facilities (word processing, spreadsheets, graphics, telecommunications, etc.) into one package. At this meeting, an overview will be provided in the general meeting, and several people will demonstrate software they use.

This meeting depends on volunteers. We need people to demonstrate the software in the classrooms (bring your own equipment).

By twisting arms, we've got a couple of demonstrators, but we need more. This is your chance to help the club, and probably to help fellow-members who might be considering the purchase of an integrated package.

If you're able to help, please notify Wayne Schaler. Without your help, this night won't happen.

April 21 **April Fools**

We've all been fooled at least once. Sometimes fooled by false advertising, false packaging, or even by an exciting sounding name which turns out to be a turkey of a program. Let's look at some of these disasters -- it may help us appreciate the good ones. Got your own horror story; bring it along. People will want to see it. Who knows; you may even be able to arrange a trade for something you do want. ==========

Musical Adventures of a Tenderfoot

by Bob Handforth

The desperate appeal for contributions, which have been voiced recently by the editors of the Newsletter, have prompted me, rather hesitantly, to submit this article for their consideration. Although a comparative newcomer to the computer hobby (I've owned my C64 less than a year) I've become deeply interested in the music and graphics features of the machine, although my original intention was to use it only for data processing. I'm still struggling valiantly with the fundamentals of the BASIC language and have not yet mustered enough courage to tackle the formidable realm of machine code.

While much of the sound and graphics software currently available is very interesting and possesses much value, I've had greater pleasure and derived more knowledge by attempting to write my own programs. Until now I have tried to take familiar songs and arrange them for the computer to play. In doing this I have usually attempted to concoct some suitable kind of graphics display, no matter how simple, to accompany the music. It somehow seemed wasteful to me to have to face a blank or idle screen while listening to the sound.

In the course of my efforts along these lines, I've "discovered" a couple of little techniques which might be worth describing. This is where I'm hesitant because these tricks might be very familiar to more experienced programmers although I've seen no mention of them in any of the several books which I have acquired.

Have you ever painstakingly written out the data for three voices in a three-part program, spent much time in editing it, and then wished you had written it all a few semitones higher? There is an easy way in which the pitch can be raised without re-writing all the data. Somewhere in your program there will probably be a statement:

```
HF%=N/256:LF%=N-256*HF%
```

where N is the note number used in the calculation. If you substituted 128 for 256 the result would give you the HF% and (very close to) the LF% values of the same note an octave higher. I have calculated the values between 256 and 128 which would produce the high and low frequencies for each of the eleven semitones within the octave, and they are as follows: (256), 243, 230, 218, 206, 195, 184, 173, 163, 154, 145, 136, (128). I have effectively used 218, for example, instead of 256 in the above statement to raise the pitch of a song by three semitones.

These numbers hold fairly true for those octaves used most frequently, namely the middle four of the eight octaves available. However, care must be taken in those areas where two or more adjacent notes have the same high frequency number, i.e., from about F# in the fourth octave and below.

If you notice a slightly off-key effect in the lowest voice, sometimes changing to an adjacent number (231, 233 or 234 instead of 232, for instance) will eliminate any distortion. In fact, any number in the range 128-256 will serve to increase the pitch of the original data, but the numbers listed above most closely approximate the eleven semitones.

One useful effect which is frequently employed in songs and musical numbers to alleviate the monotonous repetitions of lines or choruses is to raise the pitch of a repeated portion by a semitone. (A familiar example of this is the last chorus of Stevie Wonder's "I just called to say I love you".)

Since the pitch of each semitone can be raised to the next highest semitone by multiplying the frequency by the twelfth root of two (1.0595 is close enough) this can easily be done but you must have a counter variable established to record the number of sixteenth notes being played. In the common song pattern of 32 bars, the number of sixteenth notes would be 16 x 32 = 512. If this is the case then a conditional statement such as:

```
IF J>256 THEN N = N*1.0595
```

should result in raising by a semitone the pitch of the second half of the song. For added variety, the waveform can be changed in a similar manner at the same time, but the envelope of course must remain fixed.

In writing programs for the three voices to be used independently it is sometimes recommended that the first item of data for each voice be entered as "0,2". This short instant of silence allows the computer a moment to adjust to the complexities involved, and avoids a possible brief distortion at the beginning of the music.

If two of the voices are to perform in a duet of two-part harmony, a kind of guitar effect can sometimes be achieved by entering 0,3 for one voice and 0,2 for the other at the beginning of their respective data lines. This can simulate a sort of lazy, strumming sound, just as if one string of the guitar is played just before the other. Of course, if the song is to be repeated, this advance in the sound of one voice must be compensated for later in the data, so that the total combined duration value is the same for each voice.

As mentioned above, I've usually tried to have some kind of graphic display accompany the music. This isn't too difficult if the display is motionless, but there is a definite conflict with the sound if the graphic presentation is programmed to have any action or motion. On one recent effort I was able to obtain a series of flash colour changes on the screen and border which kept time to the music. However, this had the effect of causing the tempo of the music (which should have been rapid) to drop to a slow agonizing crawl.

I re-wrote the data at double speed with half of the previous duration values, and this succeeded in restoring

continued on page 14

You've Come a Long Way, Micro

by Ivan Berger

The following is from "Digital Deli", a book Scotty Adams has recommended to members at meetings. I reprint this short article because it shows the progress made since the Altair (see front cover) came out in January 1975. Reading this may be nostalgic for some, but its torture for me.

* * * * *

The horse-and-buggy days of home computing were not all that long ago. In fact, less than a decade has passed since I first read about the Altair and began thirsting -- nay, lusting -- for a computer of my own.

When I finally got one, of course, there was precious little I could do with it. My Altair turned out to be just a box with toggle switches and neat rows of red LEDs (light-emitting diodes) decorating the front panel. The switches were its only input device; the LEDs were its output, glowing and darkening in patterns that showed what you had input from the switches and what the computer had done as a result.

Both operated at the same simple-minded level as the computer itself, in binary numbers built up of 1s and 0s. A "1" was entered by raising a switch, a "0" by lowering it. The corresponding LEDs glowed for each "1" and darkened for each "0".

A one-byte program instruction, nothing more arduous than STOP, took eight switch settings.

The Plug-In Problem

Luckily, there was a world beyond the binary level. I went out and got a "terminal", with keyboard and video display screen, but then found nothing on the Altair to plug it into. What I needed was an I/O (input / output) circuit board through which the computer and terminal could communicate. The Altair had about a dozen slots where you could plug in I/O and other boards. Put a 25-pin connector into one of the oval holes in the computer's back, run a cable from the I/O board, and you had a place to plug in the terminal.

Problem: typing on the terminal had no effect on what the Altair did. Just as nice children don't speak until they're spoken to, computers don't communicate with terminals until a program has told them they can, and how to do it. There was such a program in octal (base-8) numbering that you could translate into binary and toggle in from the front panel of the Altair.

Unfortunately, the program was several hundred bytes long; at eight bits per byte, this meant well over a thousand switch-flicks before the program was in the machine. Since a computer's random access memory becomes amnesiac the instant its power is shut off, you'd have to toggle in that program every time you wanted to use the machine. And if you made just one of those thousand switch-flicks wrong,

the program wouldn't run -- or would run amok, randomly changing the bits you'd toggled in correctly.

The solution was to put that program, and any others you needed regularly, into an EPROM (erasable, programmable, read only memory), which would hold its contents even when the computer was shut off. A whole set of such programs fit easily into just two EPROM chips. Naturally the Altair had no place to plug in those chips, so I had to buy a Cromemco Byte-saver board to hold them and plug it into the Altair's bus.

With those programs, typing in instructions like C2 D5 01 ("If the zero flag is not set, jump to the program steps at memory location 01D5") was simpler than setting switch sequences of 11000010 11101010 00000001. Writing programs in tiny steps of arbitrary symbols, however, was still infuriatingly slow. What I needed was a "high-level" programming language.

By that time the Altair could run an abbreviated BASIC, the first product of a tiny new firm called Microsoft. It was only available on punched paper tape, which required a special paper-tape reader, but for once I didn't have to buy any new equipment. I had a friend with a teletype -- a printing terminal that could also read and punch such tapes. I ordered BASIC and my friend read it into his machine, then re-recorded it onto a tape cassette. I might not have a paper-tape reader, but I did have a cassette recorder.

Still Plugging Away

Not surprisingly, the Altair had no place to plug in a cassette. Now what I lacked was a board that would translate between computer and cassette deck. With this I would have a fully functioning system, including all the major parts a computer needs: a keyboard and screen (on the terminal), mass storage (the cassette), a processor, and both ROM (built-in program) and RAM (user-programmable) memory. But the Altair came with 4K of RAM memory and the BASIC interpreter took up 8K, not counting the program you then wanted to run. The solution here was to plug in still more memory boards.

In those early days, few programs were ready-made, in part because not all the programs marketed for the Altair were available in the same tape format. Besides, there simply weren't that many to choose from. Then along came the next generation of personal computers: machines like the Radio Shack TRS-80 Model I, the Commodore PET and the Apple II, with resident programs in permanent memory, cassette connections, keyboards and (except the Apple) screens built in. Not only that, but they had BASIC built in, so you could run your language immediately--no waiting while you loaded it from a tape.

In principle they were little different from my old Altair. The major difference was that these computers didn't require months of learning and searching for parts before you could use them. All you had to do was plug them in...

==========

C64 Game Design

Sixteenth of a series
by John Batchelor

Instead of writing about how to program games this month, I thought I'd add a few thoughts to the discussions about software protection, copyright and piracy which appear from time to time in this newsletter. I wish that everyone would do what I do and just release their efforts to the public domain. Vanity can be satisfied by a little claim for credit on the title screen.

As I have noted before, if only some tiny percentage of C64 owners would write a good public domain program, we will all be swamped with everything we need. Look at our disks of the month. There are now more than three dozen, each with a score of programs or more. Although some came from magazines, many are from other user groups or right here in our club. In this collection, you have word processors, data-base managers, elementary spreadsheets, music and graphics utilities, educational programs, all sorts of aids and utilities and, of course, tons of games.

How many word processors do you use regularly? I'll bet you stick with one or two at most. After a while, the marginal improvements that the latest version can add are not worth the money. People just won't pay $60 and invest hours in learning a new command structure unless a really useful new feature is offered.

Broken or pirated copies of many commercial games have become widely available over the last while. Unfortunately, there is a built-in Catch 22 with these. An easy or trivial game quickly loses its attraction and ends up at the bottom of the pile. A tough, intriguing game takes too long to play or get good at. Unless you forked out $40 for it, why spend weeks trying to master it when there are other things to do? Without an investment, there is no commitment.

When you have dozens of games, only a few will match your particular temperament and time availability. My point is that I end up using only a few programs over and over.

The best idea is for software to be "bundled" with the hardware, either by the manufacturer or the retail dealer. This is what Kaypro and Osborne did with their CP/M machines. Even Commodore provided almost free copies of all the old PET educational programs to get the C64 off the ground. Good word processing and programs such as "Sky Travel" sell the hardware.

Let Commodore package the current best word processor, data base, et cetera, with the computer. If better ones are written, start bundling them. These programs could be written on contract for the manufacturer or submitted by hopeful programmers. The authors would get a royalty based on machine sales. (Would you like a buck for every C64 sold?)

Hardware accessories such as the Koala Pad or the MIDI interfaces should come with

the current best software for them. "At cost" updates should be easily available. If everyone has a copy, there is no need for piracy and no incentive to market "me too" versions. There will be money to be made in hardware that is properly supported and in manuals and books.

To the usual arguments about disk drive damage, rights to make back-ups and so on, let me add another good reason for software to be public. I read that the manufacturer of Software Automatic Mouth (S.A.M.) is no longer in business. This program makes your C64 talk, just like a $2000 Amiga. It is (was?) cheaper than the hardware voice boxes that you can still get. My three-year-old loves this program which is patiently teaching him to make words with keyboard and screen long before he has the dexterity to print.

Copy protection on S.A.M. was not that sophisticated and programs which may use its routines are on our club disks. In theory, you would have to have a legal copy of S.A.M. before you could run programs that use its extension of BASIC. Now, with the manufacturer out of business, there is no way for you to obtain a legal version unless some dealer has old stock.

When a book goes out of print, you can usually get it from the library. You use it for a week or two and then take it back. You rent specialized tools only when you need them. You buy and keep only those books and tools that you need all the time. The same should be true for software. The most useful stuff could come with the machine. The rest should be in libraries, cheap rental outlets, down-loadable over the phone or even broadcast as FM signals.

There are other precedents where good ideas and techniques are distributed for the good of all without commercial gain. Does Dr. Christian Barnhard get $100 every time someone transplants a heart? Does George Danzig get $10 every time a linear programming problem is calculated? Do you pay anybody for the ideas on teaching or management or accounting that you read about in professional journals? No. You should give credit, but you don't pay.

Of course, the alternative is to give away the machines so that there will be a mass market for the software! There is also "shareware" where you send in $10 to the developer only if you really like the software and want documentation and support.

Or maybe, we could get companies such as Pepsi or Molson or GM to sponsor software like TV shows so long as their name appears on the screen. Pacman could eat Coke bottles instead of power pills and fruit. I wouldn't mind an ESSO logo in the corner of my spreadsheet. Put disks in with the Rice Krispies. Anyway, I hope you are convinced by now that software need not cost the earth, and that we'd all be better off if it were free, or distributed at cost.

==========

Different ways of talking to a microcomputer

by Patrick YANG

English and French are some of the "natural" languages the author can communicate in. A microcomputer, or for that matter any computer, is more limited in the sense that it can only understand machine language. To use a computer, the operator must therefore speak to it in machine language or in a more advanced form of artificial language.

Among those artificial languages is the BASIC language we all know. It is not by far the only high-level language that exists. Some are more efficient for certain tasks than others. This is not to say that BASIC is worthless. We don't always need to make the extra effort to learn another language, but it pays for those who want to perform heavy calculations, create complex data structures, have powerful control structures and subroutine interface.

First let's see briefly what sets apart machine language and a high-level language. A big step forward in those early days was the advent of assembly language that freed programmers from having to enter long strings of 0s and 1s or hard to remember hexadecimal numbers. Even then to simply add 2.4 and 2.6 would require many lines of codes. In BASIC, one would simply say A=2.4:B=2.6:C=A+B and the variable C would contain the sum. Of course, a program in machine language is much more time-efficient than the same thing written in BASIC. The problem is that any human being must make a real effort to go all the way down at the same level as the computer.

A solution is to meet half-way. A high-level language is such a compromise. A programmer will be able to write the same program in less time and be able to transport it easily from an Apple to a Mac or from an IBM PC to an Amiga at the expense of a very small loss in speed during execution.

We will quickly look at five of those high-level languages available on microcomputers:

- BASIC
- PASCAL
- LOGO
- FORTH
- C

BASIC is the most popular, and since it is often installed in ROM it will be the first one used by many users. A simple "Hello" on the screen can be obtained by a:

```
10 PRINT "Hello"
RUN <CR>
```

PASCAL will need a little more persuasion than that:

```
PROGRAM first;
BEGIN
  WRITELN(' Hello');
END.
```

LOGO will require:

```
TO first
  PRINT [Hello]
END
```

FORTH will do it with:

```
: first
  ." Hello"
;
```

In C, it would look like:

```
#INCLUDE "STDIO.H"
MAIN()
{
PRINTF("%s"," nHello n");
}
```

Very different ways to achieve exactly the same thing. The power of these languages lies not in the similarities, but in the differences that exist between them.

BASIC, LOGO and to a lesser extent, FORTH are usually available as interpreted languages on a microcomputer. That means that the program will be translated into machine language one line at a time and each time you execute it. PASCAL and C, on the other hand are compiled languages. After the program is entered, you must have a compiler translate, once and for all, the code in machine language. You will then execute a program much faster than if it had been interpreted.

The all-time winner for heavy computation is FORTRAN, without a doubt, but C will do a very good job. It is very low-level among the high-level languages and its speed can be impressive.

Data structures are pieces of information that are related, for example in a directory each subscriber to Bell Canada could be entered in a record. PASCAL would create each entry by:

```
TYPE entry = RECORD
     name: STRING[25]
     adress: STRING[79]
     number: STRING[9]
  END;
VAR subscriber: ARRAY [1..x]
of entry;
...
subscriber[400].name := 'A B
Smith'
subscriber[400].adress := '123
Main St'
subscriber[400].number :=
'123-4567'
```

As you can see PASCAL is very powerful for that type of application.

Control structures are ways of choosing one path or another in a program, depending on a given condition. IF cond THEN [ELSE] is available in all languages in one form or another. C and PASCAL are, among the five described here, the big winners for the number of control structures. They use very similar constructions:

CASE:choose from many alternatives
FOR:counted loops
WHILE cond:conditional loops
REPEAT...UNTIL cond:conditional loops

A subroutine is a portion of a program that is used many times. Instead of copying it many times, calling the subroutine from the main program saves memory and time. BASIC will use GOSUB line number. All the other languages presented here can call a subroutine by its name, as in LOGO:

```
TO SQUARE
  REPEAT 36 [ SQUARE RIGHT 10
]
END
```

Some even have private variables and global, like C and PASCAL. Private variables will only be accessible from within that subroutine and can even have the same name as another variable in the program:

```
MAIN()
{
     INT a;
     a=2;
     PRINTF("\n%D",a);
     sub1();
     PRINTF("\n%D",a);
}
void sub1()
{
     INT a;
     a=9;
     PRINTF("\n%D",a);
}
```

As you have probably guessed by now, you would see 2, 9 and 2 being printed on your screen. A global variable on the other hand would have given you 2,9 and 9.

A last thing to consider when you have to choose a high-level language is its user-friendliness. An interpreted language is considered more user-friendly than a compiled one. You can type a program and by executing it still get a result in a matter of seconds. LOGO is the best illustration of such a language.

The purpose of this article was to get people interested in going beyond BASIC. It was not intended to frighten you away with all those different syntaxes. If it did, please consider the advantages of understanding what other magazines are talking about.==========

==========

MINUTES -- JANUARY MEETING

by Bob Mason

The meeting was held on Monday January 20, at Charlebois High School. Paul Irwin opened the meeting with an announcement regarding the Club Bulletin Board. The new phone number for the board will be 723-2252. Unfortunately, that's about all we've got right now. A deal was arranged but fell through, and another deal will have to be arranged. If anyone knows of a large-capacity drive (1 meg or more) for sale, give Don White a call.

Scotty talked about the *Print Shop Users Newsletter* and *Print Shop Companion*, two publications to help users of Broderbund's "Print Shop". (Contact Scotty for information regarding these).

Scotty also talked about Nostalgia Night and April Fools Night, and Paul added his plea for member participation in Nostalgia Night.

Telecom Canada had a team of speakers at the meeting, and a video presentation, slides and overheads. And even though they seem to have made some moves to better accommodate intelligent, as well as dumb terminals, interest in the presentation seemed to drop off when the price of the service was mentioned. But several members stayed on to discuss iNet 2000 afterwards.

For a report on 9-10 p.m. activities, see notes in the "Agenda" article.

==========

MUSIC *from page 7*

the tempo to a suitable level. However, in the process I also lost all my sixteenth notes and this adversely affected the quality of what I thought was otherwise a fine effort. It's apparently much easier to slow down the tempo by using loops than it is to increase it. If anyone has an easy formula for increasing the tempo without a complete re-write, I'd be glad to hear from them.

Occasionally in our Disks of the Month there will be interesting programs dealing with sound or graphics which can be listed for examination. This permits greenhorns like myself to study and analyze them, acquire new ideas, and discover new lines of thought.

Such programs often provide a better source of instruction and are more informative than many of *COMPUTE's* lengthy compilations (in which I have found errors), or the elaborately-designed software packages which have a high entertainment value but which are low on education.==========

RAMBLINGS
by Rob Mason

John Cocklin started a good thing last month with this column, and I'd like to continue it. Anyone else is welcome to contribute as well.

GOOD DEALS

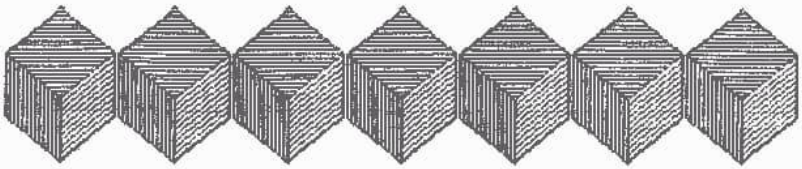
As last month, this is by no means an exhaustive list, but only a few specials I've noticed over the past couple of months.

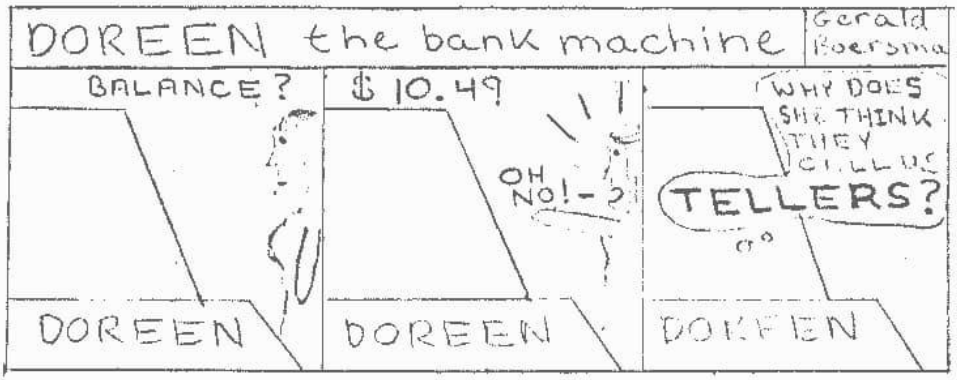
1. K-Mart (of all places) has picked up a bundle of old Electronic Arts software at $9.95 a piece. Of the titles, I only remember Hard Hat Mack and D-Bug. These are older titles, but the company is one with a reputation for a good product.

Record companies have "deletes", book publishers have "remainders", and now software publishers too find it cheaper to dump a product than keep an inventory on it. We win for once.

2. A Florist & Computers, 1152 Ogilvie Road, is just that -- a florist and a computer store combined, and it is a pleasant combination. They deal mainly with Atari, but do sell and rent Commodore software, and printers and supplies which work with any system. Rental prices and membership are under review, but will probably remain competitive.

They also have a supply of old Aardvark Software adventure games for $3.95 each. They're not sophisticated by today's standards, but Aardvark was one of the pioneers way back when, and probably still worth a look.





MACCOM SYSTEMS LTD.

PERSONAL-COMPUTERS SOFTWARE HARDWARE
DISKETTES & SUPPLIES

Specializing in software/hardware
and accessories for

COMMODORE
IBM
APPLE MAC

Visit " Computers for Le$$ "
at 43 William St. (233–3919)
1 block North of
Rideau Centre

HOW TO SHOP AT MACCOM

749-1232

BY PHONE

Join **TPUG** the World's largest Commodore users' group

- Free information-packed **Monthly Magazine**
- Access to 4000 **Public Domain Programs**
- Annual **Convention**
- Member **Bulletin Board**
- For $30.00 ($20.00 for students), all the above and attend the many TPUG meetings in the **Toronto Area**

* * * * * * * * * * * * *

SPECIAL RATE FOR Ottawa 6502 members!!!

Contact Don White for details.

* * * * * * * * * * * * *